GETOPT

Vulnerable to internal buffer overflows

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-23

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 4826 bytes

Attack Category	• Malicious Input
Vulnerability Category	• Buffer Overflow • Input source (not really attack) • Unconditional
Software Context	• String Parsing
Location	
Description	Some implementations of getopt() are vulnerable to internal buffer overflows. The getopt(int argc, char *const argv[], const char *optstring) function is used to parse the command line parameters. The level of security risk is implementation dependent, in that for some C packages, it is possible for a buffer overflow to occur. The third argument of the function is a list of option characters. If the option character is followed by a colon, then the option requires an argument; two consecutive colons means the argument is optional. This is used to specify options such as -w in the command line. Flag instances of getopt(), getopt_long(). Look for nearby bounds checks. There is a portability issue for old platforms. Check documentation for your particular platform.
APIs	**Function Name** / **Comments** getopt / certain old OS versions are flawed getopt_long / certain old OS versions are flawed getopt_long_only / certain old OS versions are flawed
Method of Attack	Some version of getopt() allow overflow of the internal buffer. This allows an attacker to overflow the buffer by providing a really long command line

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

	parameter. This can potentially lead to arbitrary code execution. The risk is greatest for setuid programs.
Exception Criteria	
Solutions	<table><tr><th>Solution Applicability</th><th>Solution Description</th><th>Solution Efficacy</th></tr><tr><td>Whenever getopt or getopt_long is used and changing library versions is not an option.</td><td>Check the bug reports for your implementation of C and verify that the inputs are within acceptable lengths before calling the function.</td><td>Effective if accurate bug reports are available and correct checks are done.</td></tr><tr><td>Whenever getopt or getopt_long is used.</td><td>Upgrade to a version of the C library that does not have a vulnerable getopt implementation.</td><td>Effective. Preferred solution.</td></tr></table>
Signature Details	int getopt(int argc, char * const argv[], const char *optstring); int getopt_long(int argc, char * const argv[], const char *optstring, const struct option *longopts, int *longindex);

Examples of Incorrect Code	<pre>while ((c = getopt (argc, argv, "abc:")) != -1) {
	/* do something with option */
	}</pre>

Examples of Corrected Code	<pre>/* Ensure argument sizes are within platform-specific bounds. */
	
	if (argc > MAX_SAFE_ARGC) exit(1);
	for (i = 0; argc>i; i+
	+) if (strlen(argv[i]) > MAX_SAFE_ARGLEN) exit(1);
	
	while ((c = getopt (argc, argv, "abc:")) != -1) {
	/* do something with option */
	}</pre>

Source Reference	• Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X , pg. 148.

Recommended Resource	
Discriminant Set	**Operating System**: Windows **Languages**: C, C++

Cigital, Inc. Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. mailto:copyright@cigital.com